Adaptive Software for a Changing World

Assessing the state-of-the-art in Dynamic Discovery of Ad Hoc Network Services

Christopher Dabrowski, Olivier Mathieu, Kevin Mills,
Doug Montgomery, and Scott Rose

ARDA Review Meeting
July 18, 2001

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **18 JUL 2001**	2. REPORT TYPE	3. DATES COVERED **00-00-2001 to 00-00-2001**

4. TITLE AND SUBTITLE **Assessing the state-of-the-art in Dynamic Discovery of Ad Hoc Network Services**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **National Institute of Standards and Technology,Gaithersburg,MD,20899**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

ARDA Review Meeting, 18 Jul 2001

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**37**	

Presentation Outline

- Introduction to Dynamic Discovery Protocols
  - Project Motivation
  - Goals & Technical Approach
  - Generic Model Encompassing (Five) Discovery Protocols
- Modeling & Analysis
  - Architecture-based Approach
  - FY01 Accomplishments
- Measurement
  - Measurement Methodology
  - Synthetic Workload Generation
  - FY01 Accomplishments
- Project Plans for FY02
- Supplemental Material (included but not presented)
  - Includes list of papers, software artifacts, and presentations
  - Includes sample results from modeling, analysis, & measurement

What is a dynamic discovery protocol?

Dynamic discovery protocols enable **network elements** (including software clients and services, as well as devices):

(1) to **discover** each other without prior arrangement,

(2) to **express** opportunities for collaboration,

(3) to **compose** themselves into larger collections that cooperate to meet an application need, and

(4) to **detect and adapt to changes** in network topology.

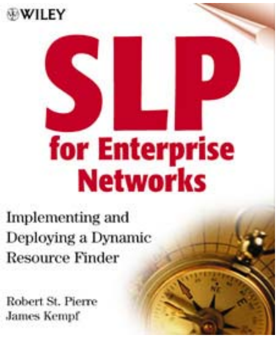
Why analyze dynamic discovery protocols?

- In the future, **all software systems will be distributed** systems written to operate over a network, where conditions vary.
- Dynamic **discovery protocols provide a foundation** upon which such distributed systems will be constructed.
- **Understanding** the current (first) generation of discovery protocols **essential** to enable government agencies to establish requirements and to help industry to improve designs for the second and subsequent generations.

Selected Current (First) Generation Protocols for Dynamic Service Discovery

3-Party Design

2-Party Design


WILEY
SLP for Enterprise Networks
Implementing and Deploying a Dynamic Resource Finder
Robert St. Pierre
James Kempf

Adaptive 2/3-Party Design

Vertically Integrated Design

Network-Dependent Design

Network-Dependent Design

Our Goal

To provide **metrics** and approaches to **compare and contrast emerging** dynamic **discovery protocols**, **to better understand** their critical functions, to identify weaknesses, and to strengthen the quality and correctness of designs for future protocols.

Our General Technical Approach

- Build a **generic, domain model** (UML) providing consistent terminology encompassing a range of service discovery protocols.
- Build **executable models** of service discovery protocols from extant specifications, and analyze them under conditions of dynamic change.
- Build **measurement infrastructure** and measure implementations of dynamic service protocols for scalability.
- Build **simulation models** of dynamic service protocols and assess the adaptability of such models to dynamic change.
- Design, model, and evaluate **protocol mechanisms** that enable discovery protocols to self-adapt in the face of dynamic change *(this part of the project is funded by the DARPA Fault Tolerant Networks program).*

Technical Approach to Modeling & Analysis

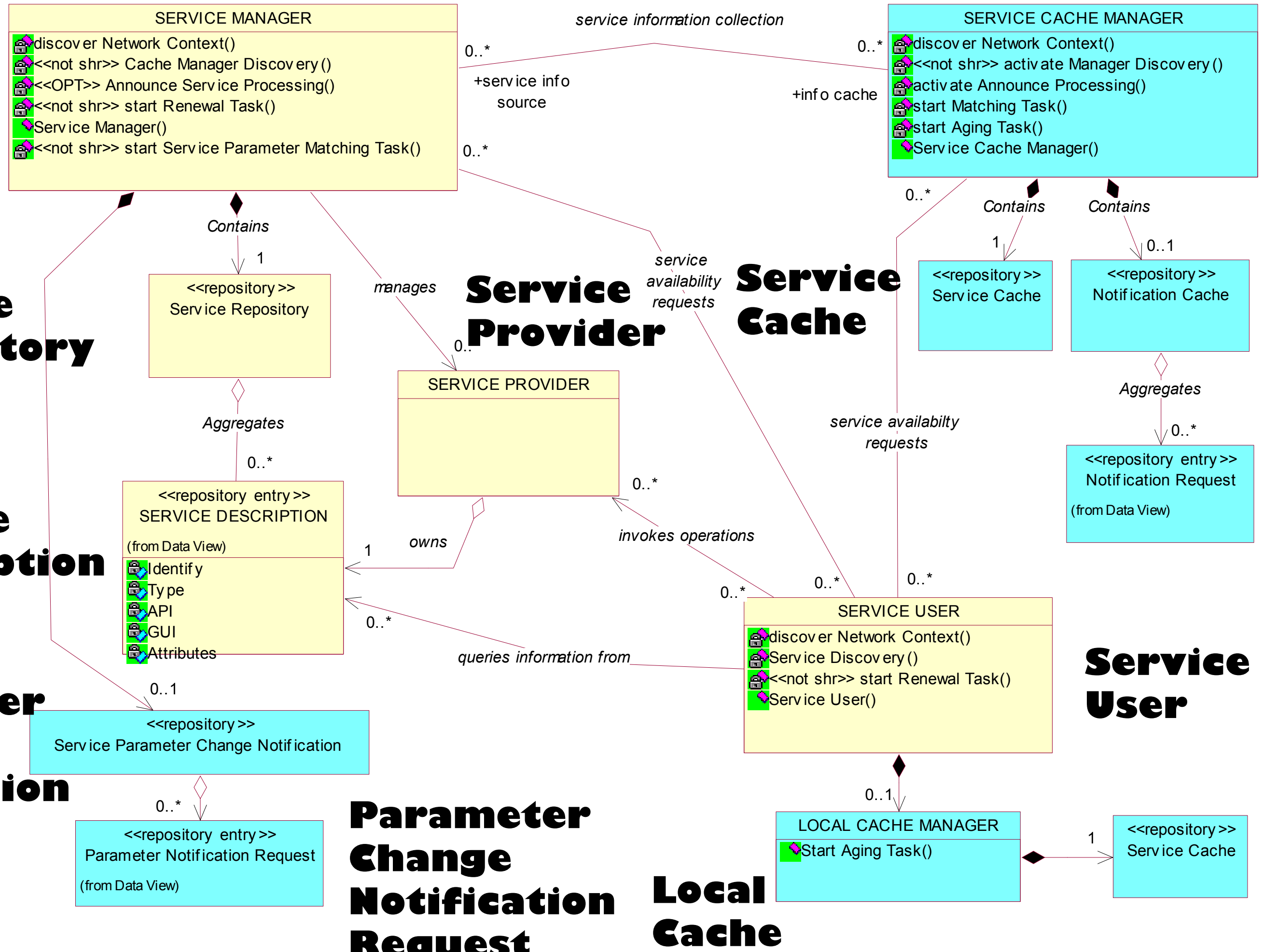
- **Model** Discovery Protocol **specifications using Architectural Description Languages** (ADLs) and associated tools
- **Analyze** Discovery Protocol **models** to assess consistency, correctness, and completeness under conditions of dynamic change.
- **Compare and contrast** our **models** with regard to function, structure, behavior, performance, complexity, and scalability under conditions of dynamic change.

Technical Approach to Measurement

- Design technology-independent **benchmark service and scenarios**.
- Create **synthetic workload generation tools** for emulating the behavior of large-scale dynamic ad hoc networking environments.
- Develop implementation-independent **performance measurement methodologies and tools** for service discovery protocols (SDPs) and required supporting protocols.

ITL Pervasive Computing Portfolio

Foundation for Comparisons: A Generic Structural Model (UML) for Service-Discovery Domain



Service Manager

Service Cache Manager

Service Repository

Service Provider

Service Cache

Notification Cache

Service Description

Notification Request

Service User

Parameter Change Notification Cache

Parameter Change Notification Request

Local Cache Manager

Local Service Cache

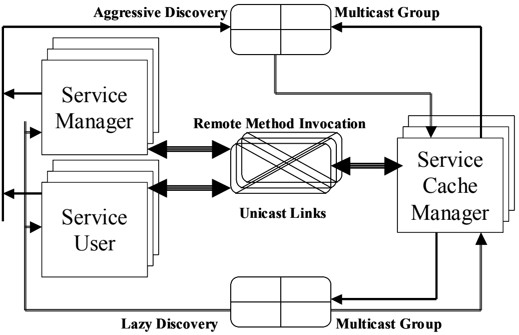
Architecture-based Approach to Modeling and Analysis

(using Rapide, an Architecture Description Language and Tools Developed for DARPA by Stanford)

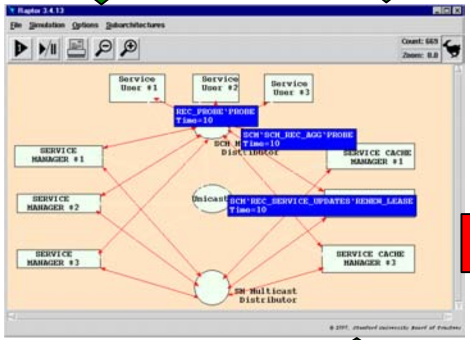
Time	Command	Parameters
5	NodeFail	SM4
5	LinkFail	SCM1 SM4
10	GroupJoin	SM4 GROUP1
10	FindService	SU8 5 1 2 S XYZ ALL
50	AddService	SM4 SCM3 T ATT API GUI 20 30

Scenario

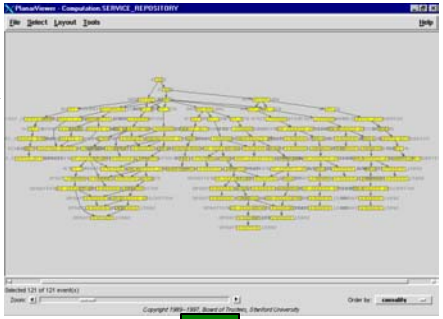
Topology



Specification Model



Execute with Rapide


PlanarViewer - Computation SERVICE_REPOSITORY
File Select Layout Tools
Help
Selected 121 of 121 event(s)
Zoom:
Order by: causality
Copyright 1989–1997, Board of Trustees, Stanford University

```
-- ****************************************************
-- ** 3.3  DIRECTED DISCOVERY CLIENT INTERFACE      **
-- ****************************************************
-- This is used by all JINI entities in directed
-- discovery mode.  It is part of the SCM_Discovery
-- Module. Sends Unicast messages to  SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need reviw.
TYPE Directed_Discovery_Client
  (SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
   InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR    : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES      : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update   : DD_SCM_Update;
SERVICE SCM_Update      : SCM_Update;
SERVICE DB_Update       : dual DB_Update;
SERVICE NODE_FAILURES : NODE_FAILURES;  -- events for failure and recovery.
ACTION
 IN Send_Requests(),
    BeginDirectedDiscovery();
BEHAVIOR
   action animation_Iam (name: string);
   MySourceID         : VAR IP_Address;
   PV              : VAR ProtocolVersion;
```

Consistency Conditions

For All (SM, SD, SCM):
(SM, SD) IsElementOf SCM registered-services (CC1)
implies SCM IsElementOf SM discovered-SCMs

For All (SM, SD, SCM):
SCM IsElementOf SM discovered-SCMs & (CC2)
(SD) IsElementOf SM managed-services
implies (SM, SD) IsElementOf SCM registered-services

For All (SM, SD, SCM):
SCM IsElementOf SM discovered-SCMs & (CC3)
(SM, SD) IsElementOf SCM registered-services &
NOT (SCM IsElementOf SM persistent-list)
implies Intersection (SM GroupsToJoin, SCM GroupsMemberOf)

For All (SM, SD, SCM, SU, NR):
(SU, NR) IsElementOf SCM requested-notifications & (CC4)
(SM, SD) IsElementOf SCM registered-services &
Matches((SM, SD), (SU,NR))
implies (SM, SD) IsElementOf SU matched-services

Analyze POSETs

Assess Correctness, Performance, & Complexity

Real-Time Checking of Consistency Conditions

Sample Consistency Condition*

For All (SM, SD, SCM): (SM, SD) *IsElementOf* SCM registered-services
implies SCM *IsElementOf* SM known-SCMs

…that is, a Service Manager should register its Services on an Service Cache Manager *only if* it maintains that Cache Manager on its "known SCM" LIst.

*Assuming absence of network failure and normal delays due to updates

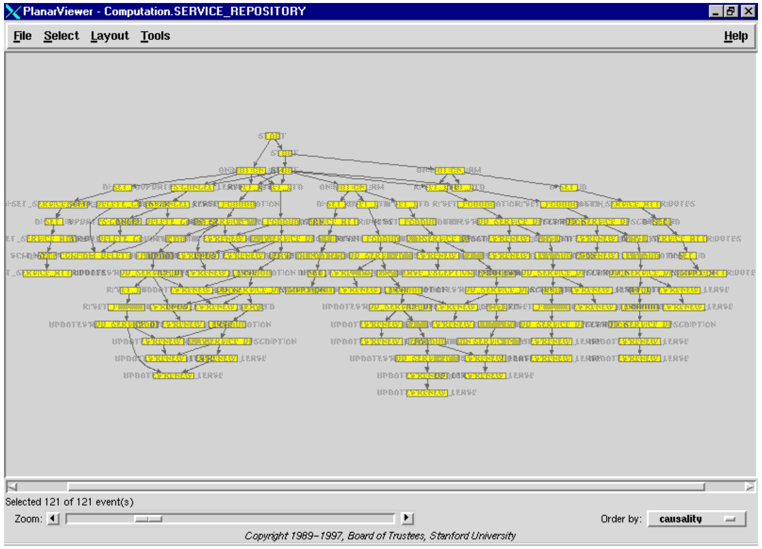
- SM is Service Manager
- SD is Service Description
- SCM is Service Cache Manager

- registered-services is a set of (SM,SD) pairs
- known-SCMs is a set of SCMs

Same executable model can be used to assess selected performance properties and to measure complexity+

+future work on the project intends to investigate the relationship between design complexity (applying ideas from Kolmogorov Complexity) and design quality (as represented by violation of consistency conditions)

Off-Line Analysis of POSETs to Compare and Contrast Behaviors Given a Congruent Topology and Scenario



Selected 121 of 121 event(s)

Order by: causality

Copyright 1989–1997, Board of Trustees, Stanford University

Performance Metrics

- Message volume?
- Message intensity?
- Discovery latency?
- Query latency?
- Recovery latency?

Complexity Metrics

- *Description length of models (static and dynamic forms)?*
- *Compressibility of message exchanges?*
- Cyclomatic measure of models?
- Time-Space computational complexity?

Quality Metrics

- Count of consistency violations?

POSET analyses provide basis for defining *metrics* that provide quantitative measures of properties of a modeled system

Modeling & Analysis: FY01 Accomplishments

- Developed an **initial UML model** of a generic service discovery protocol with specific projections for Jini, UPnP, and SLP *(the three most general discovery protocols)*
- Developed and exercised an **executable Rapide model of Jini**, including selected consistency conditions and performance metrics. *(Proving our concept, with the exception of complexity metrics, which require us to complete at least a second model.)*
- Briefed Jini modeling at PC2001, and provided Jini designers with **results of analysis** *(see examples in the supplemental material).*
- Wrote **paper**, "Analyzing Properties and Behavior of Service Discovery Protocols using an Architecture-based Approach", and submitted it to an upcoming *Conference on Complex and Dynamic Systems Architecture*

Anticipated Additional FY01 Accomplishments

- Develop an **executable Rapide model of Universal Plug-and-Play** (UPnP) subset
- Devise some **performance scenarios** against which to compare UPnP and Jini
- Define possible **complexity and quality metrics** to use in comparing discovery protocols

Measuring Performance and Scaling of Service Discovery Protocols

- **What are the performance requirements for SDPs?**
  - First generation SDPs designed around consumer applications.
    - 10's of consumer electronic devices in the home.
    - 100s-1000s of shoppers walking about.
    - 10,000s-? cars on a highway.
  - **Requirements for DoD applications?**
- **Scale**
  - Number of nodes, services, directories, and clients?
  - Size of PC network topologies?
- **Dynamic Range**
  - Rate of service/client arrival departure?
  - Service load of advertisements, queries, control, events?
  - Latency requirements for service discovery?
- **Network Technologies**
  - Range of link technologies?
  - Performance of WAN connections?

Measurement Methodology

- **SDP Architectures & Protocols Differ Considerably**
  - Directory based vs flat peer-to-peer (combinations)
  - Java RMI and Serialized Objects vs. HTTP/SOAP/GENA and XML

- **How to reasonably compare the performance of such diverse technologies?**

- Usage-Based **Scenarios & Metrics**
  - Service initiation – restart, auto configuration, advertisement, renewal
  - Client active query – restart, by name, single instance, multiple instances, all instances
  - Client passive monitoring – persistent query for new instances, network restart
  - Event Notification and control – registration latency, notification latency, distributed control performance (control + event notification).

- SDP-Independent **Benchmark Service –**
  - Simple counting device/service that can be used to exercise all significant discovery/control capabilities of Jini and UPnP.
  - Supports – get/set service, GUIDs/attributes/type, control, event notification, GUI

- Implementation-Independent **Measurement Tools**.
  - Measure on-the-wire
  - Response/Load

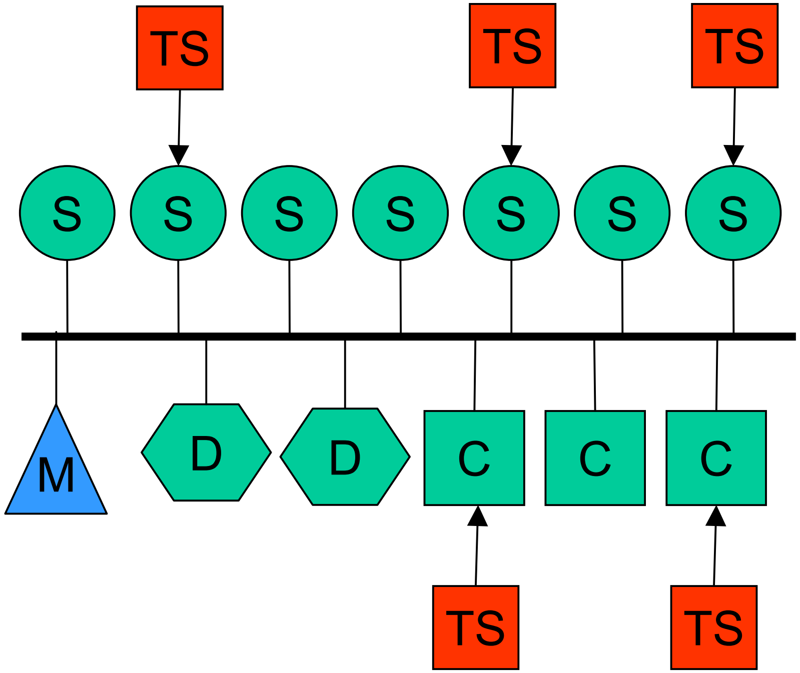
Synthetic Workload Generation Tools

- **Objective** – Emulate large, dynamic environments of 100’s of devices/services and 10’s of control points / clients.
  - – Dynamic devices providing the benchmark service.
  - – Scripted control points execute measurement scenarios.

- **SDP Experimenters Toolkits**
  - –Drive real SDP implementations
  - –Emulate the behavior of a large number of dynamic devices
  - –Emulate the behavior control points/ scripted behavior for testing

– **Jini & UPnP Initial development complete**

- – SunMS Jini, Intel UPnP on Linux platforms.
- – Target of 100’s of devices and 10’s of control points met.



SDP Performance & Scalability : FY01 Accomplishments

- **Methodology and scenarios** for comparative evaluations of SDPs
- **Synthetic workload generation/emulation tools** for Jini and UPnP.
- **Performance measurement tools** for SDPs and supporting protocols.
- **Initial studies** of client query scenarios *(see examples in the supplemental material).*

Anticipated Additional FY01 Accomplishments

- Begin to **expand test bed** with 802.11 and NIST Net emulated WAN links
- **Complete** event notification & control **studies**
- **Publish** complete set of **performance results** for Jini and UPnP

FY02 Project Plans

Assessing Function, Structure, and Behavior

- Compare and contrast Jini and UPnP *(requires us to complete UPnP model)*
- Investigate relationships among complexity, quality, and performance
- Develop model of Service Location Protocol
- Update and extend generic UML model to include messages and behavior

Assessing Performance, Scalability, and Dynamic Range

- Expand test bed with 802.11 LANs and NIST Net emulated WAN Links
- Incorporate Service Location Protocol implementation into measurements
- Initiate research into other aspects of dynamic range, scaling, and fault tolerance.
- Develop large-scale simulations for study of SDP behavior in highly dynamic, ad hoc network environments.

Supplemental Material

- Papers, Software Artifacts, & Presentations
- Interactions & Impact
- Other DoD Programs Related to this Project
- Individual Quad Charts for Modeling & Analysis and Measurement Portions of the Project
- Sample Modeling & Analysis Results
- Sample Measurement Results

Papers

- Christopher Dabrowski and Kevin Mills, "Analyzing Properties and Behavior of Service Discovery Protocols using an Architecture-based Approach", submitted to *Working Conference on Complex and Dynamic Systems Architecture*, to be held December 2001.

- Olivier Mathieu, Doug Montgomery, Scott Rose, "Empirical Scaling Analysis of Service Discovery Technologies", *work in progress.*

Software Artifacts

- Generic UML Model (in Rational Rose format) of Discovery Protocols, including specific projections to Jini, UPnP, and SLP

- Rapide Model of Jini

- *Jini and UPnP Experimenter's Toolkits* consisting of synthetic workload generation tools, scenario scripts, and performance measurement tools for SDPs.

Presentations

- Christopher Dabrowski, "Applying ADLs to Assess Emerging Industry Specifications for Dynamic Discovery of Ad Hoc Network Services", *Dynamic Assembly for System Adaptability, Dependability, and Assurance (DASADA),* DARPA PI Meeting, January 31, 2001

- Christopher Dabrowski and Kevin Mills, "Modeling and Analysis of Service Discovery Protocols", *Pervasive Computing 2001*, May 2, 2001, Gaithersburg, Maryland.

- Olivier Mathieu, Doug Montgomery, Scott Rose, "Empirical Measurements of Service Discovery Technologies", *Pervasive Computing 2001*, May 2, 2001, Gaithersburg, Maryland.

Interactions & Impact

- Christopher Dabrowski and Kevin Mills met for three hours with Jim Waldo, SUN's Jini Architect, to discuss the NIST-developed Rapide model of Jini and the approach to analysis

- Subsequently, Jim Waldo reviewed and commented on the paper that Chris and Kevin wrote describing their architecture-based approach to modeling and analysis of service discovery protocols

- Our modeling and analysis of Jini uncovered several areas in the Jini natural-language specification that could be improved.

- Empirical measurements of Linux UPnP stack led to detection and correction of errors in scaling mechanisms (i.e., jitter algorithms). NIST staff collaborated with Intel Architecture Labs to correct the implementation of the Linux UPnP jitter algorithms.

- Other issues with respect to the impact of static parameter tuning on UPnP scaling were identified. Future research will examine the use dynamic control algorithms to address these issues.

Other DoD Programs Related to this Project

- Fault Tolerant Networks – DARPA Program
  http://www.darpa.mil/ito/research/ftn/index.html

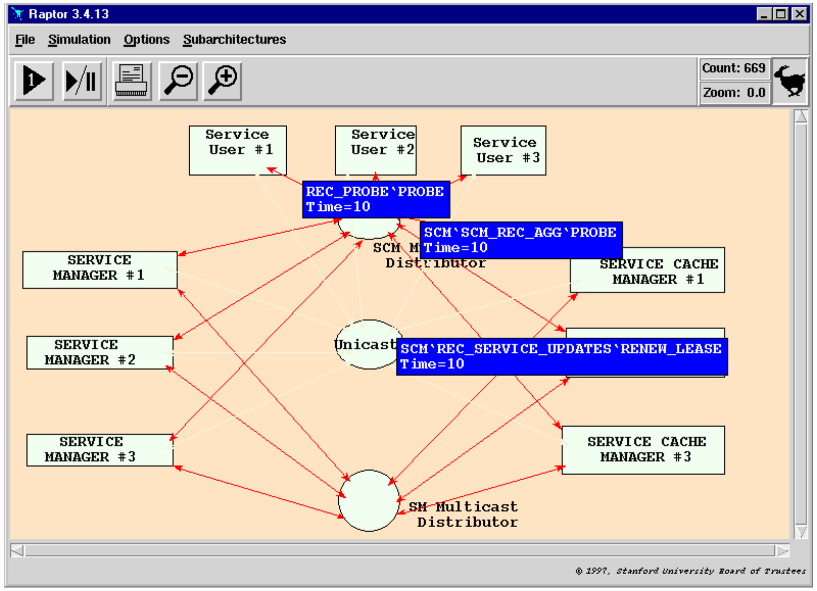
- OpenWings – Joint Motorola-Sun-U.S. Army Program (key enabler of Joint Vision 2020)
  http://www.openwings.org/index.htm

- Organically Assured and Survivable Information Systems (OASIS) – DARPA Program
  http://www.darpa.mil/ato/programs/oasis.htm

- Dynamic Assembly for System Adaptability, Dependability, and Assurance (DASADA) - DARPA Program
  http://www.darpa.mil/ito/research/dasada/index.html

- Critical Infrastructure Protection (CIP) and High Confidence, Adaptable Software (SW) Research Program of the University Research Initiative (URI) – Office of Naval Research
  http://www.onr.navy.mil/sci_tech/special/cipswuri/

Model & Analyze SDP Function, Structure, and Behavior



Count: 669
Zoom: 0.0

REC_PROBE`PROBE
Time=10

SCM`SCM_REC_AGG`PROBE
Time=10

SCM`REC_SERVICE_UPDATES`RENEW_LEASE
Time=10

© 1997, Stanford University Board of Trustees

Objectives

(1) Provide increased understanding of the competing dynamic discovery services emerging in industry
(2) Develop metrics for comparative analysis of different approaches to dynamic discovery and assuring quality and correctness of discovery protocols
(3) Assess suitability of architecture description languages to model and analyze emerging dynamic discovery protocols

Technical Approach

- Develop ADL models from selected specifications for service discovery protocols and develop a suite of scenarios and topologies with which to exercise the ADL models
- Propose a set of consistency conditions & constraints that dynamic discovery protocols should satisfy
- Propose a set of metrics, based on partially ordered sets, with which to compare and contrast discovery protocols
- Analyze ADL models to assess consistency condition satisfaction, and to compare and contrast protocols

Recent Accomplishments:

- Developed a generic UML model encompassing the structure and function of Jini, UPnP, SLP, Bluetooth, and HAVi
- Projected specific UML models for Jini, UPnP, and SLP
- Completed a Rapide Model of Jini structure, function, and behavior
- Drafted and implemented a scenario language to drive the Rapide Jini Model.
- Developed a set of consistency conditions and constraints for Jini behavioral model; currently being tested using scenarios.
- Discovered significant architectural issue in interaction between Jini directed discovery and multicast discovery

Products & Contributions

- Rapide specifications of Jini, Universal Plug and Play (UPnP), and Service Location Protocol (SLP)
- Scenarios and topologies for evaluating discovery protocols
- Suggested consistency properties for service discovery protocols
- Suggested metrics, based on partially ordered sets (POSETs), for comparing and contrasting discovery protocols
- Paper identifying inconsistencies and ambiguities in service discovery protocols and describing how they were found
- Paper proposing consistency conditions for service discovery protocols, and evaluating how Jini, UPnP, and SLP fare
- Paper comparing and contrasting Jini, UPnP, and SLP at the level of POSET metrics

Measure SDP Performance, Scalability, & Dynamic Range

Objectives

(1) Provide a quantitative, comparative analysis of the performance and scaling characteristics of emerging service discovery protocols (SDPs).
(3) Design methodologies and tools for performance and scaling measurement of SDPs and supporting protocols.
(4) Develop simulation tools for large scale ad-hoc network / application environments

Technical Approach

- Design and develop experimenters toolkits for conducting live performance analysis of SPDs implementations.
- Propose metrics and scenarios for comparing the performance of multiple SDP protocols.
- Design and develop simulation models of emerging SDPs and adhoc network environments.
- Analyze and compare the performance of SDPs based upon testbed measurements and simulation.

Recent Accomplishments:

- Designed methodology and scenarios for comparative performance evaluation of live Jini and UPnP implementations.
- Established testbed with Jini, UPnP implementations.
- Developed synthetic workload generation tools for Jini and UPnP capable of emulating 10's-100's of devices/services and control point / clients.
- Discovered scaling problems with Linux UPnP 1.0 implementation. Conducted initial investigations in protocol / parameter tuning to increase the scalability of this implementation.
- Began design and development of on-the-wire performance measurement tools for SDPs and supporting protocols.

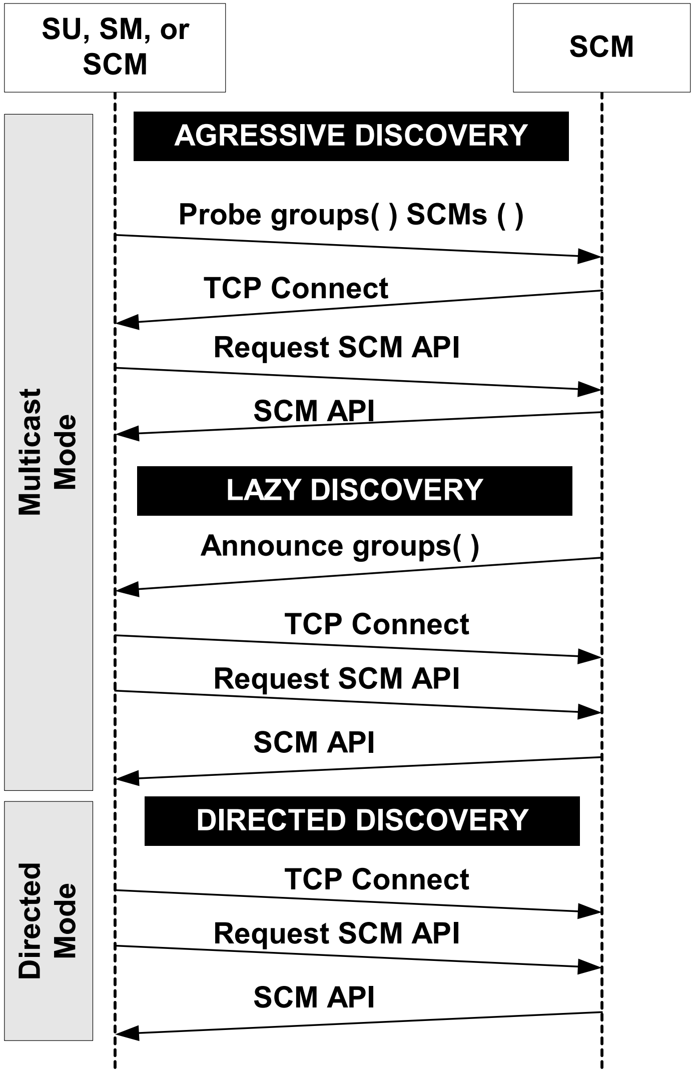
Products & Contributions

- Experimenter's toolkits consisting of synthetic workload generation tools, scenario scripts, and performance measurement tools for SDPs.
- Measurement methodologies and tools for SDPs and supporting protocols.
- Ad-hoc network simulation environment and SDP protocol models.
- Publications / standards contributions providing quantitative analysis of the relative performance and scaling properties of SDPs.

Sample Modeling & Analysis Results

Jini Has Two Discovery Modes Encompassing Three Discovery Processes

This is not a result, but background needed to understand the next few slides.



SU, SM, or SCM

SCM

Multicast Mode

AGRESSIVE DISCOVERY

Probe groups( ) SCMs ( )

TCP Connect

Request SCM API

SCM API

LAZY DISCOVERY

Announce groups( )

TCP Connect

Request SCM API

SCM API

Directed Mode

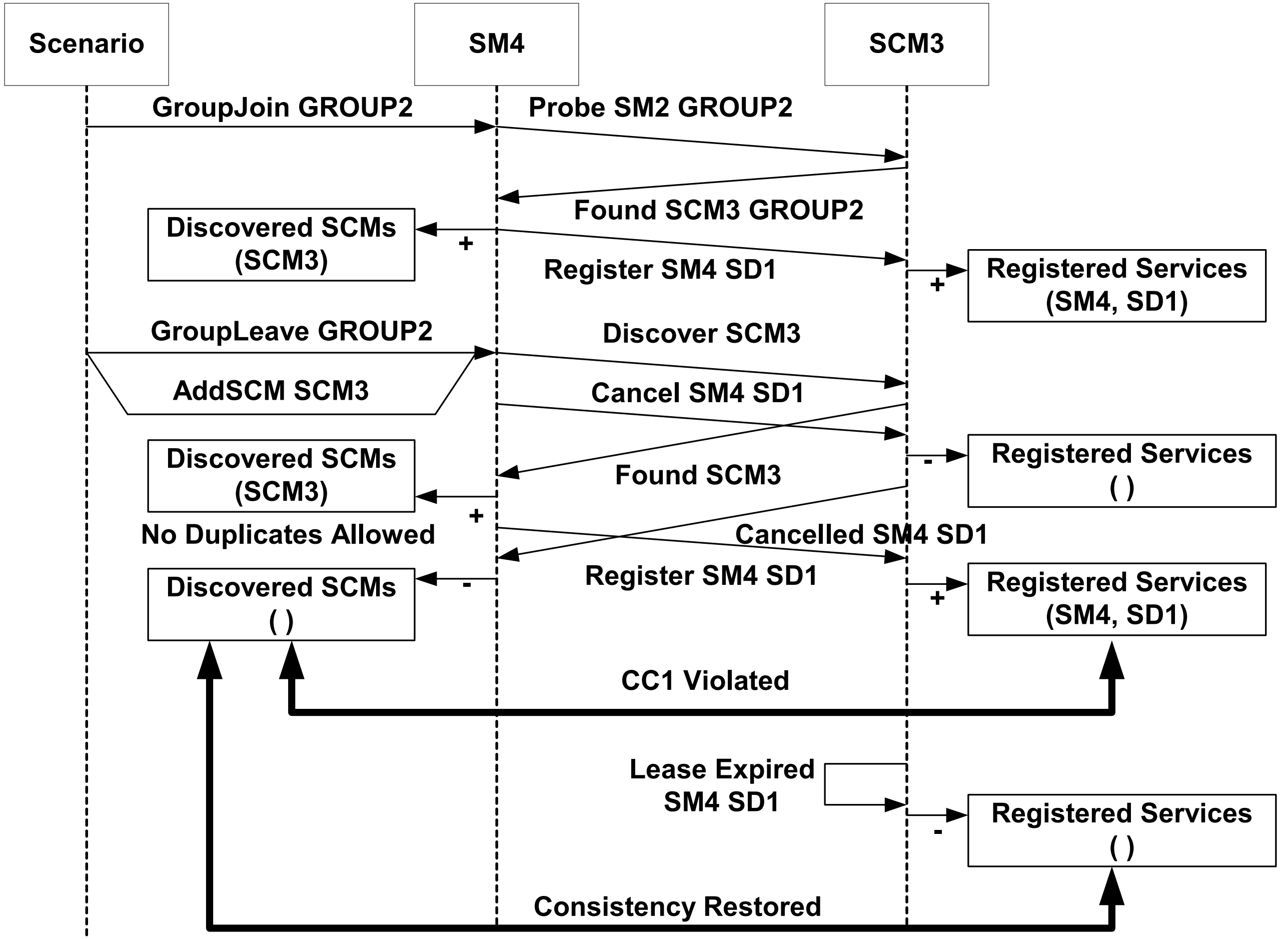
DIRECTED DISCOVERY

TCP Connect

Request SCM API

SCM API

Local Interference between Directed and Multicast Discovery

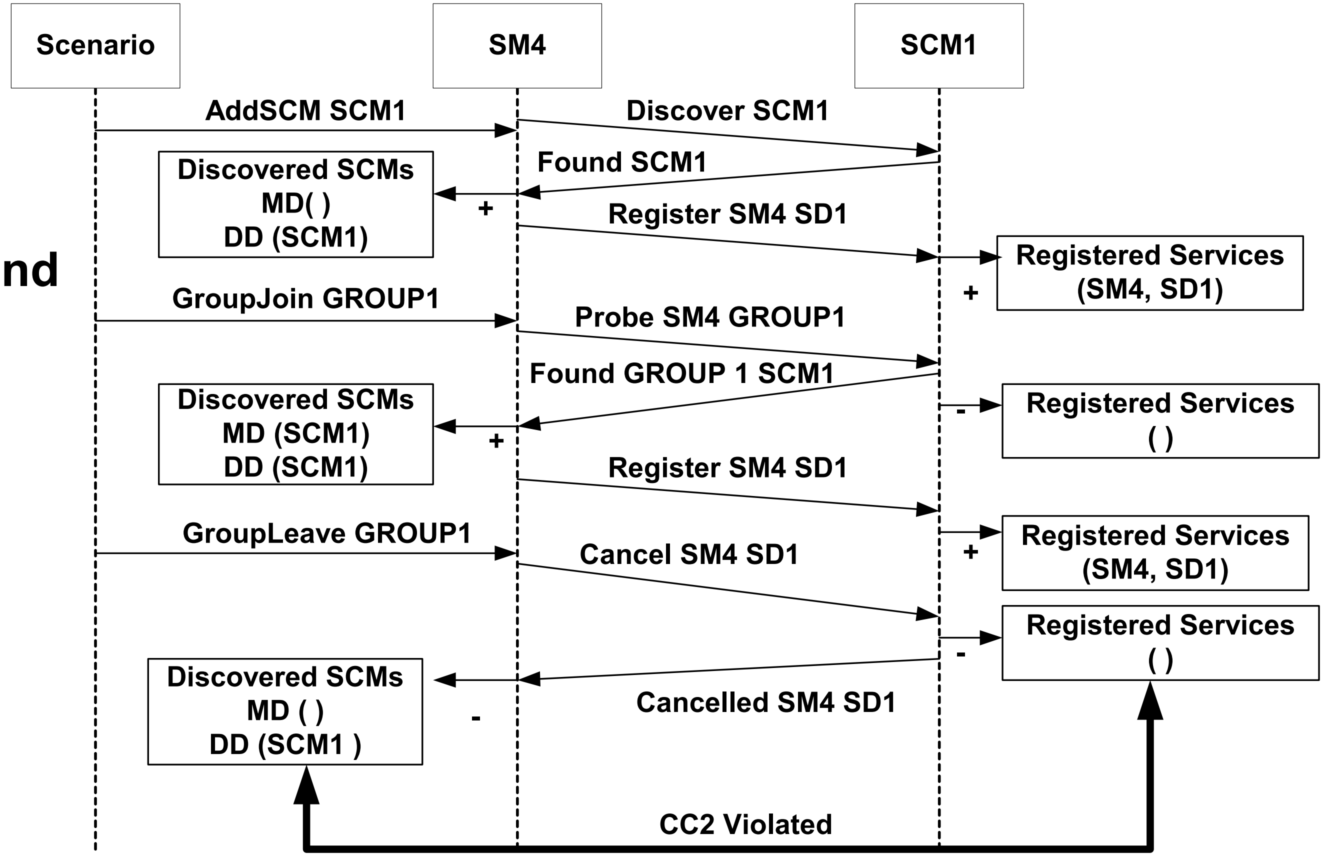


Based on one possible interpretation of ambiguous specification

For All (SM, SD, SCM):
(SM, SD) IsElementOf SCM registered-services (CC1)
implies SCM IsElementOf SM discovered-SCMs

Remote Interference between Directed and Multicast Discovery

Based on a second possible interpretation of ambiguous specification

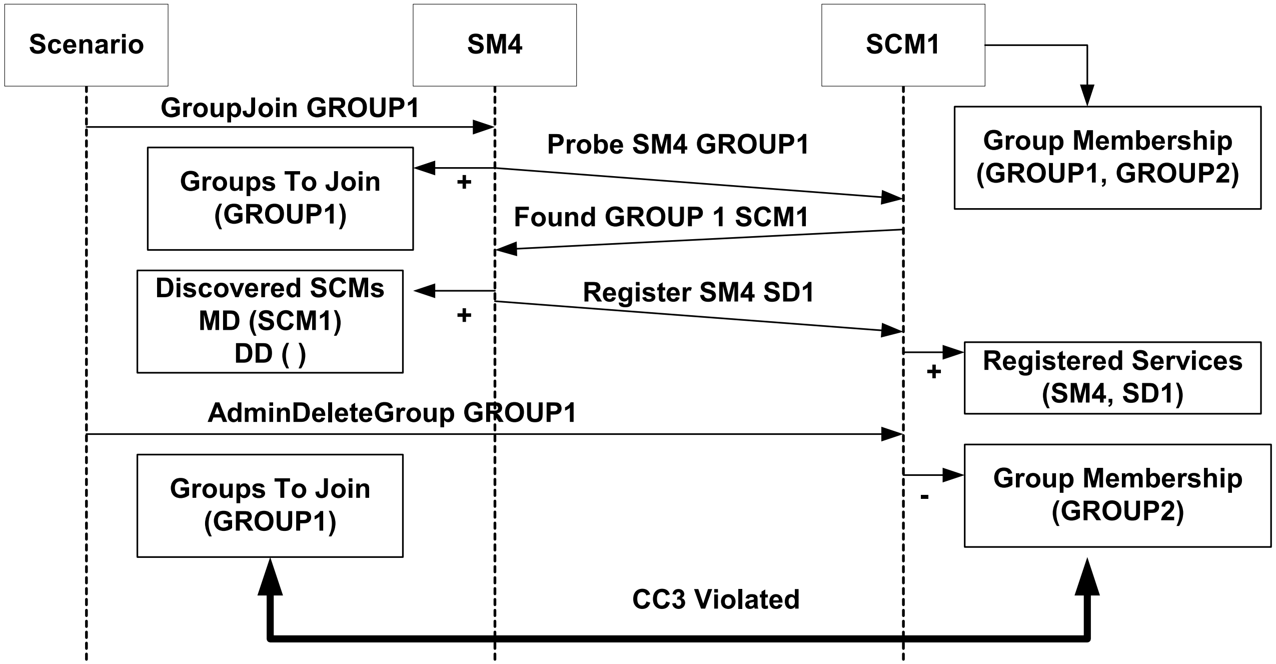


For All (SM, SD, SCM):

SCM IsElementOf SM discovered-SCMs & (CC2)
(SD) IsElementOf SM managed-services
implies (SM, SD) IsElementOf SCM registered-services

Insensitivity to Dynamic Changes in Group Membership by SCM

Specification incomplete with regard to this issue

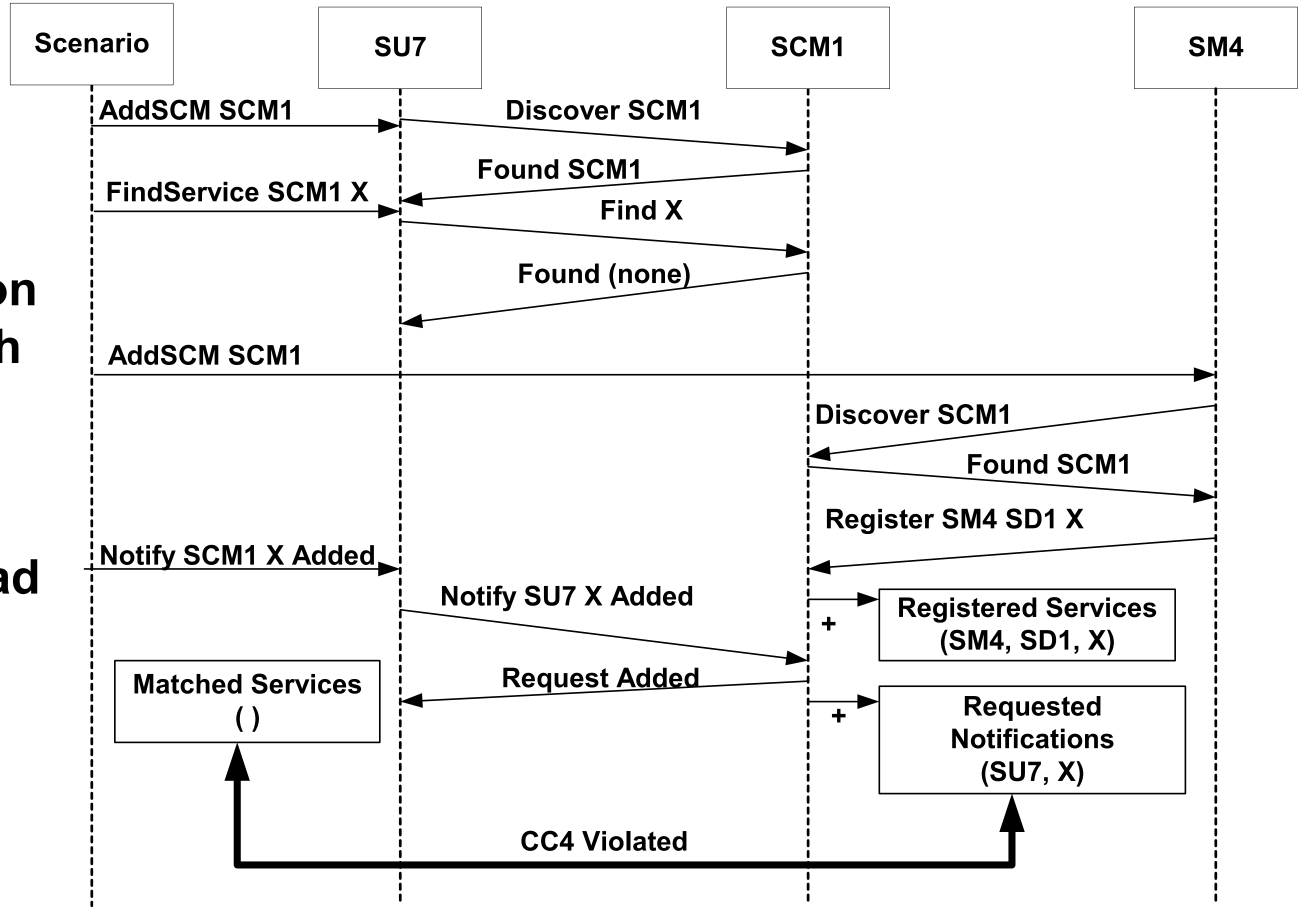


For All (SM, SD, SCM):

SCM IsElementOf SM discovered-SCMs & (CC3)
(SM, SD) IsElementOf SCM registered-services &
NOT (SCM IsElementOf SM persistent-list)
implies Intersection (SM GroupsToJoin, SCM GroupsMemberOf)

A Registration Race Condition

Specification does not mention this issue, which could catch application programmers unaware and lead to unexpected behavior



For All (SM, SD, SCM, SU, NR):
(SU, NR) IsElementOf SCM requested-notifications & (CC4)
(SM, SD) IsElementOf SCM registered-services &
Matches((SM, SD), (SU,NR))
implies (SM, SD) IsElementOf SU matched-services

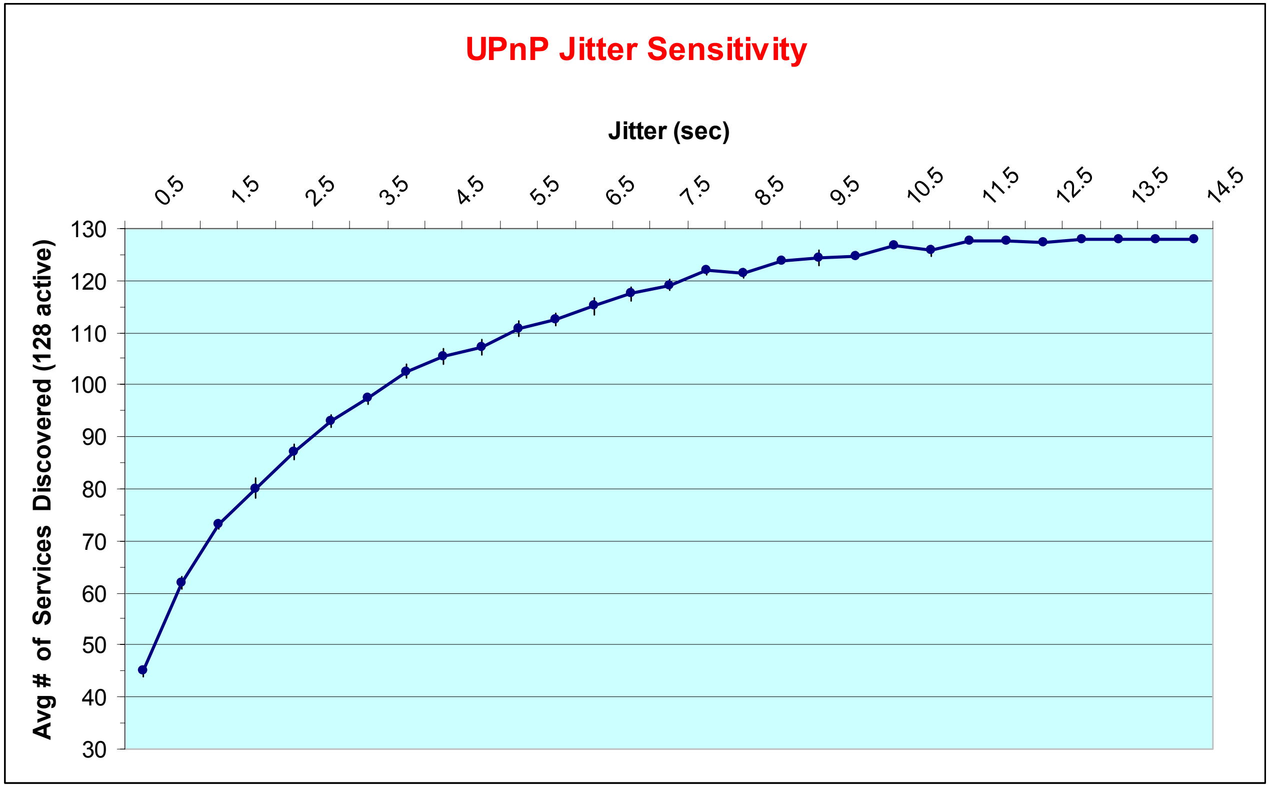
Sample Measurement Results

Notes:

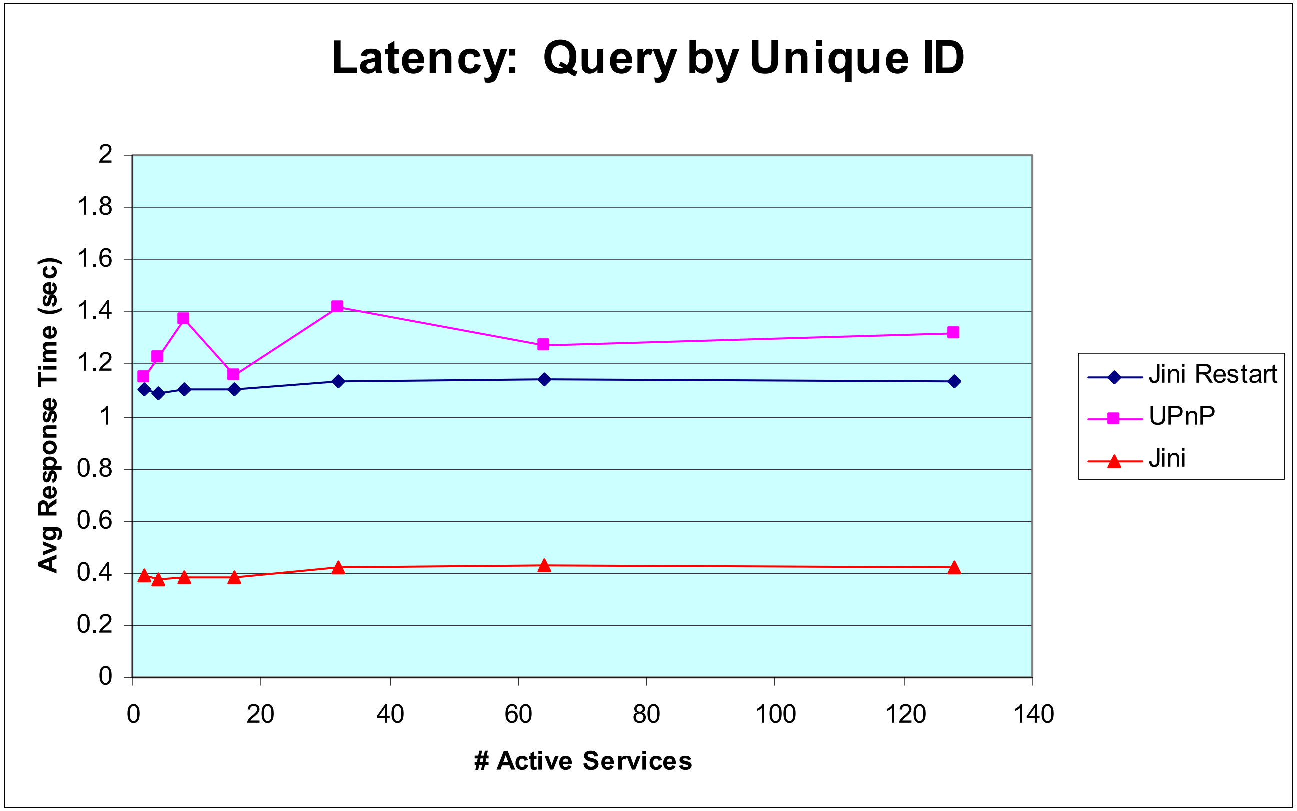
- Example results of Client Active Query scenarios.
- Query experiments measure latency / load through the client download of the service proxy / description.
- Results show two scenarios for Jini
  - Jini Restart – includes overhead of client discovery and first access of look up server.
  - Jini – query from “warm” client, after lookup server Discovery has completed.
- Default UPnP jitter value is 2.5 seconds
  - In device poll experiment, jitter values of 5.5 sec (64 services) and 11 sec (128 services) were used.

Linux UPnP Scaling Problems

- Problems encountered in achieving initial scaling goals for device emulation tools.
  - UPnP scalability above 40 devices a function of protocol tuning parameters (e.g., response jitter, multicast retransmission factor).
  - Errors in implementation of jitter algorithms

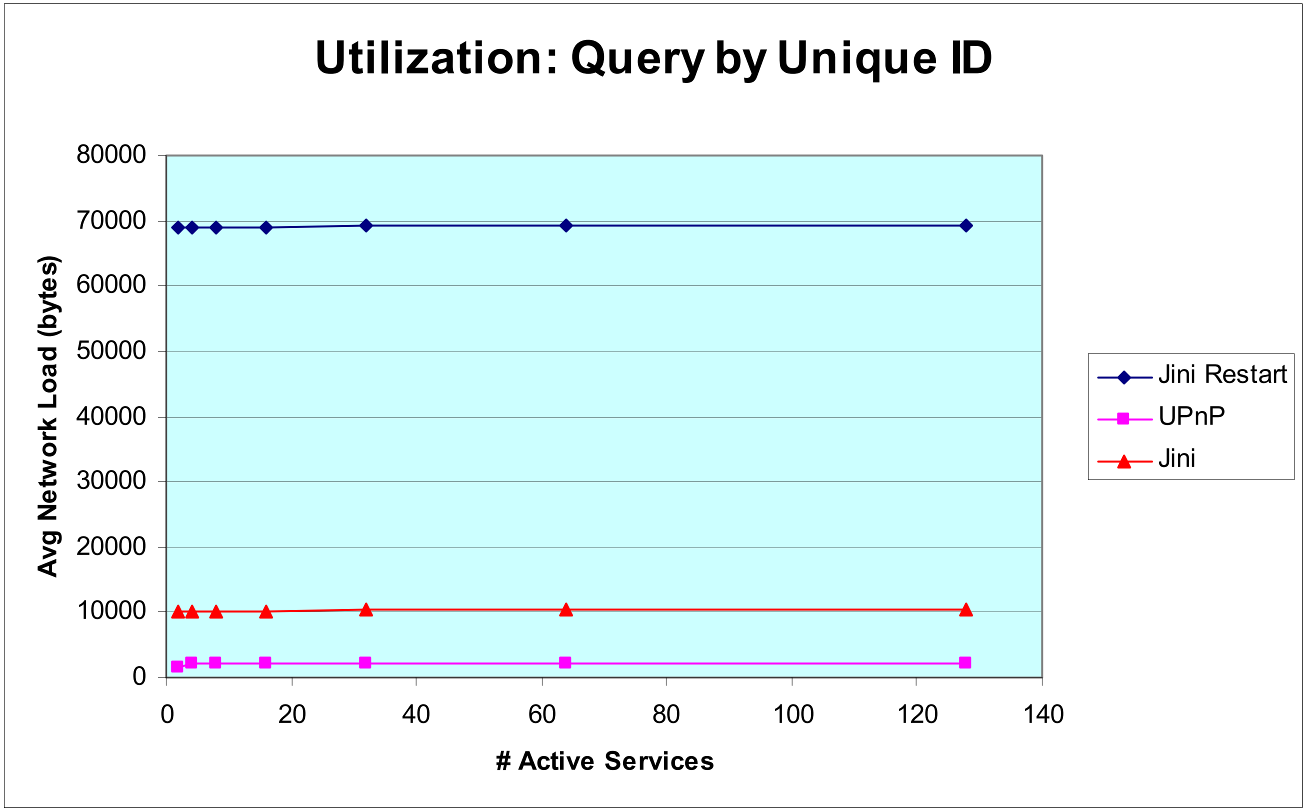


Some Example Results: Jini vs UPnP Discovery



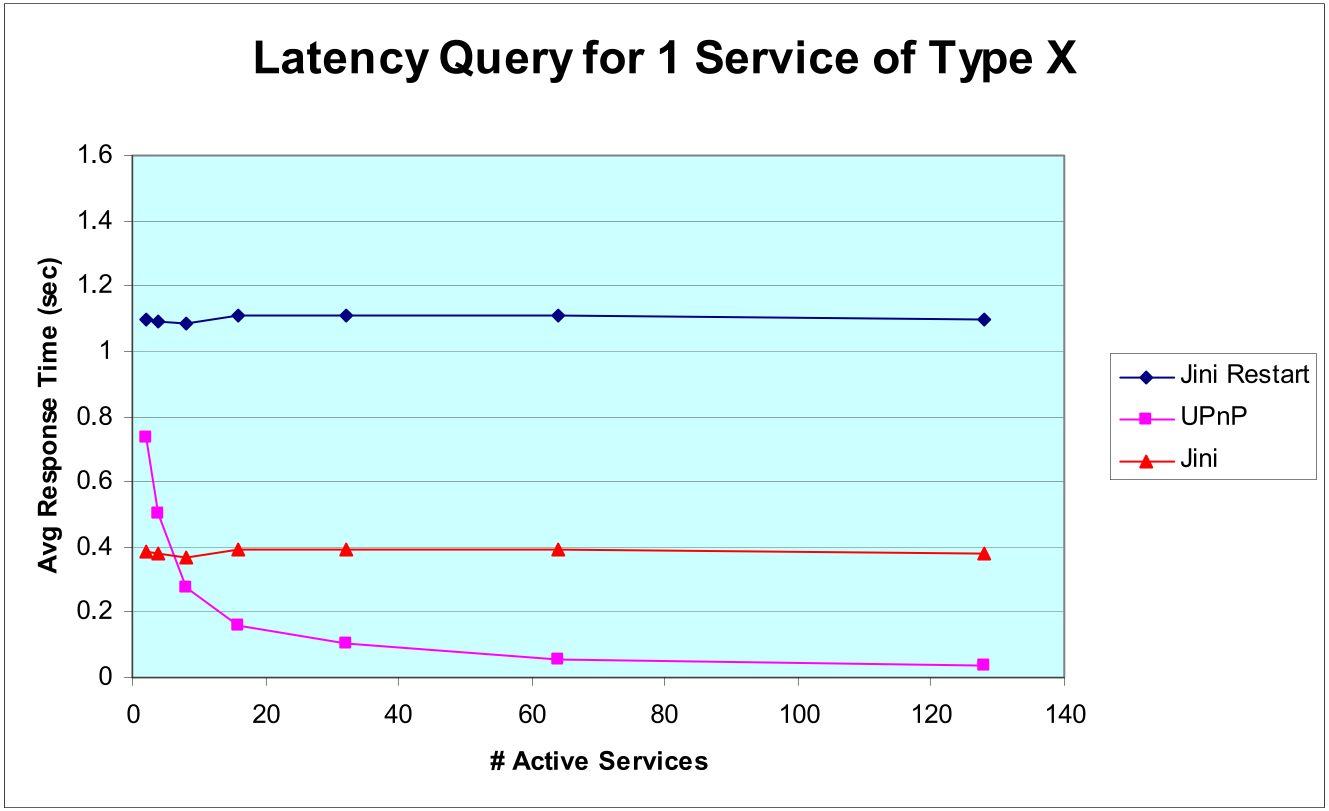
Performance of Query for Specific Service: "Find service X"

Some Example Results: Jini vs UPnP Discovery



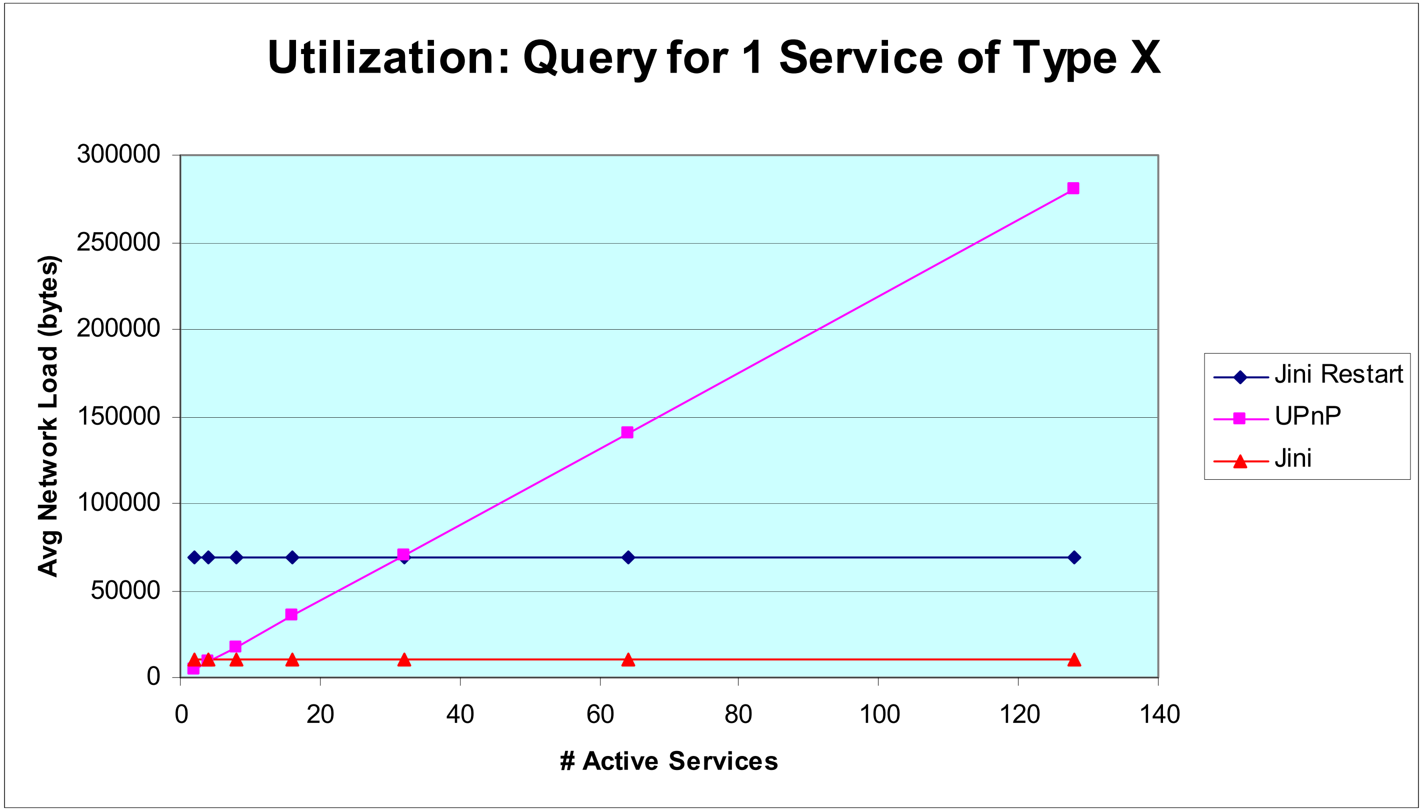
Performance of Query for Specific Service: “Find service X”

Some Example Results: Jini vs UPnP Discovery



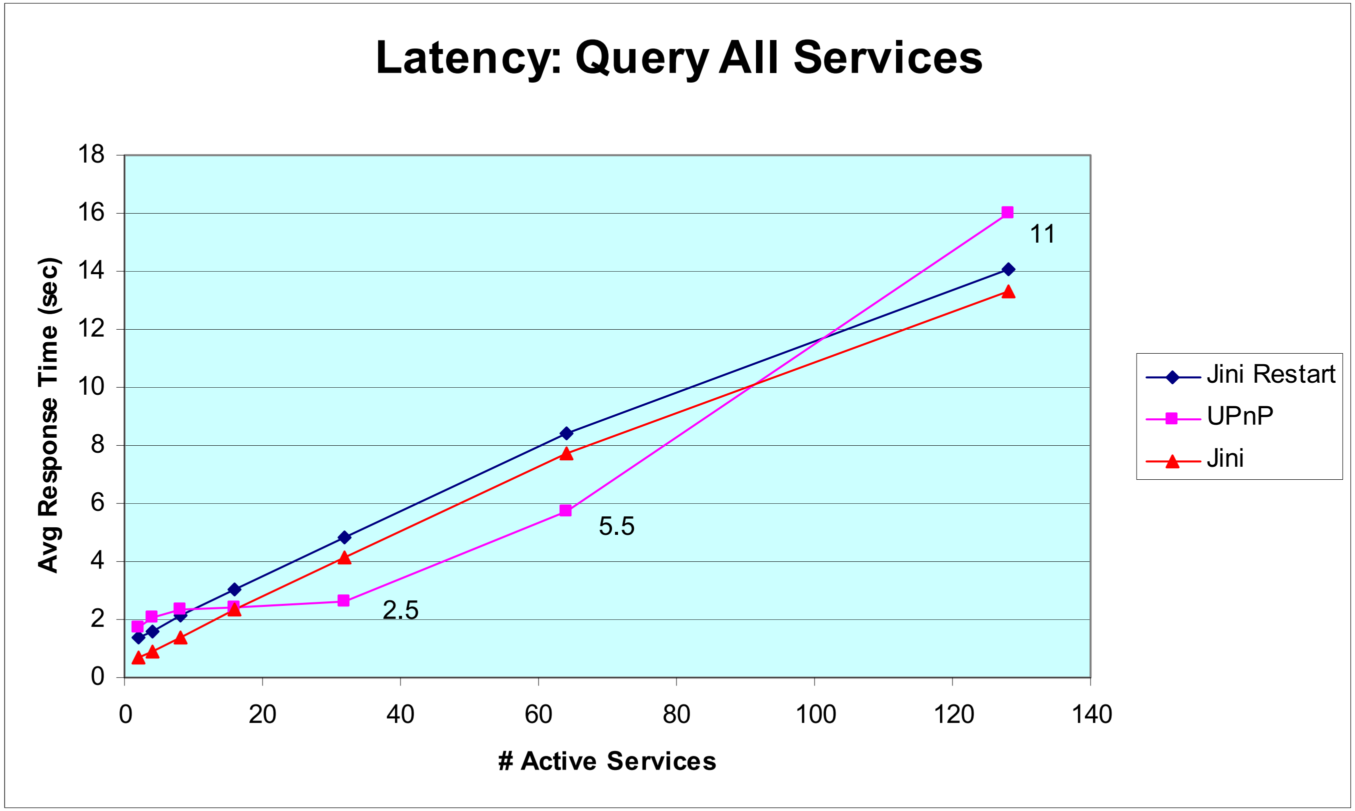
Performance of “anycast” Query: “Find one instance of type X”

Some Example Results: Jini vs UPnP Discovery



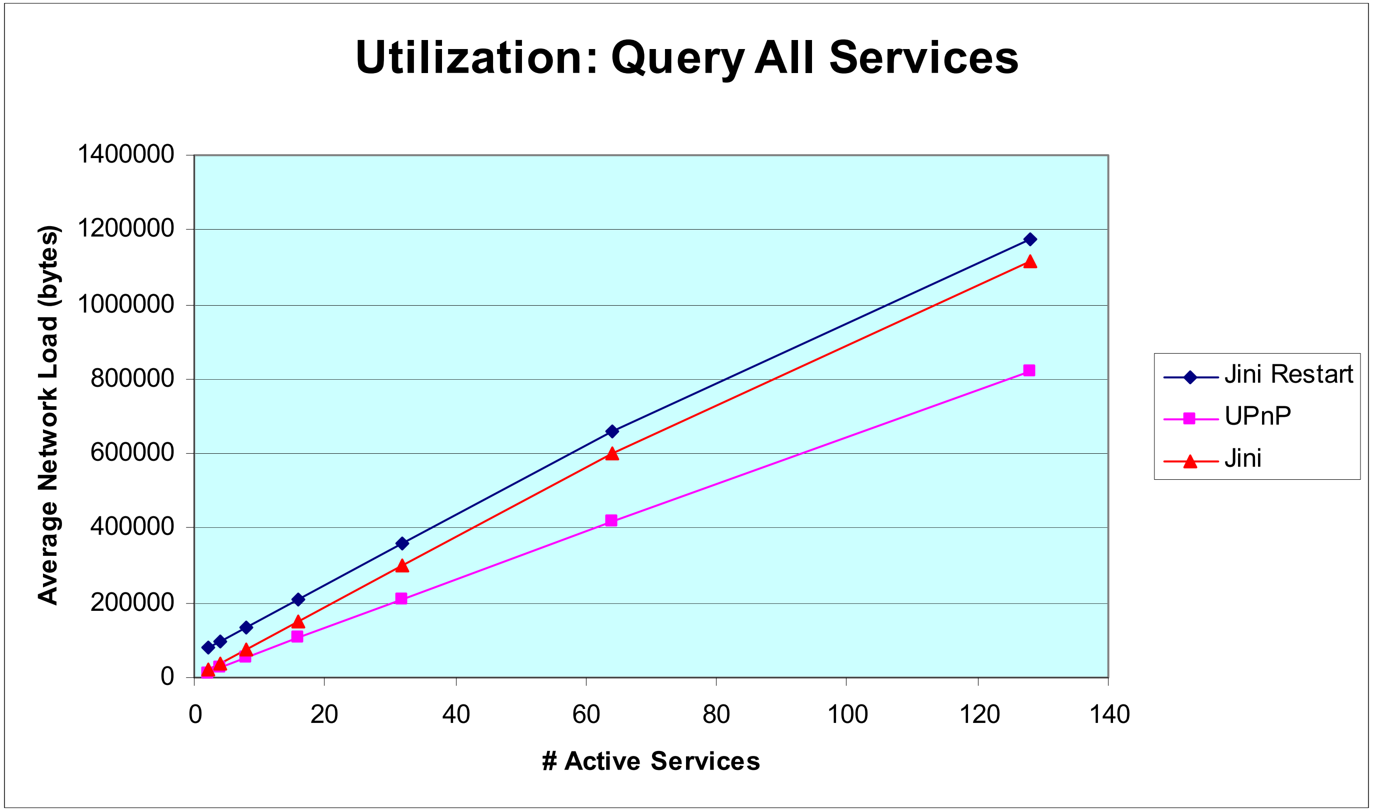
Performance of “anycast” Query: “Find one instance of type X”

Some Example Results: Jini vs UPnP Discovery



Performance of Service/Device Poll: "Find all active services"

Some Example Results: Jini vs UPnP Discovery



Performance of Service/Device Poll: “Find all active services”